

CIRCUIT DESIGN AND CONFIGURATION FOR LOW POWER FPGAs

A Dissertation Proposal by

Seyi Ayorinde

17 February 2015

Submitted to the graduate faculty of the
Charles L. Brown Department of Electrical and Computer Engineering
in partial fulfillment of the requirements
for the Dissertation Proposal and
subsequent Ph.D. in Electrical Engineering

Approved By:

Scott Acton

John Lach

Kevin Skadron

Ben Calhoun

Steve Bowers

Outline

Abstract	iii
1 Introduction	1
1.1 Motivation for Low-Power FPGAs - Ubiquitous Computing	1
1.2 Motivation for Configuration Scheme for Custom-FPGAs	3
1.3 Thesis Statement	4
1.4 Goals	4
2 Background	5
2.1 General FPGA Architecture	5
2.2 Commercial Ultra-Low Power FPGAs	6
2.3 Academic Ultra-Low Power FPGAs	8
3 Circuit Exploration	10
3.1 Motivation	10
3.2 Prior Art	10
3.3 Research Question	11
3.4 Approach	11
3.5 Evaluation Metrics	15
3.6 Results	17
3.7 Anticipated Results	23
3.8 Contributions	23
4 Architecture Exploration	25
4.1 Motivation	25
4.2 Prior Art	25
4.3 Research Question	26
4.4 Approach	26
4.5 Evaluation Metrics	27
4.6 Results	28
4.7 Anticipated Results	29
4.8 Contributions	30
5 Toolflow	32
5.1 Motivation	32
5.2 Prior Art	32
5.3 Research Question	33
5.4 Approach	33
5.5 Results	36
5.6 Anticipated Results	37
5.7 Contributions	38
6 Embedded FPGAs in SoCs	39
6.1 Motivation	39
6.2 Prior Art	39
6.3 Research Question	40
6.4 Approach	40
6.5 Evaluation Metrics	41
6.6 Anticipated Results	42
6.7 Contributions	42
7 Research Tasks and Schedule	43

8 Publications	44
8.1 Completed	44
8.2 Planned	44

Abstract

Today, current technology is moving towards the idea of ubiquitous computing (UbiComp), where sensors and other integrated circuits (ICs) are ever present in daily life, creating a smart environment for people to constantly monitor and react to their environments. For UbiComp to be fully realized, a host of requirements are necessary for the ICs deployed in this vast network. These requirements include low cost, low power consumption, flexibility, and adequate computation power. Current IC design methodologies do not quite meet all UbiComp requirements. Application Specific Integrated Circuits (ASICs) are low power and have the necessary computing power for UbiComp applications, but they are prohibitively inflexible. Ultra-Low Power (ULP) General Purpose Processors (GPPs) have the necessary flexibility, but are prohibitively high power (by multiple orders of magnitude). Field Programmable Gate Arrays (FPGAs) are another IC implementation that could potentially be used for UbiComp applications, because it bridges the gap between the high efficiency of the ASIC and the high flexibility of the GPP. However, FPGAs have historically been targeted for high performance applications, where performance (or speed) is the main metric. As a result, commercial FPGAs are too power hungry for ULP applications (like UbiComp). Relatively little research has been done to bring reconfigurable logic into the ULP application space. However, if FPGAs can be retargeted for ULP applications, then they will intrinsically bridge that same gap, this time between ULP ASICs and ULP GPPs, by providing energy efficiency than ULP GPPs, but also have the flexibility to update or re-target their applications (a necessary requirement for UbiComp) without the expensive, time-consuming respins that ULP ASICs require.

This document proposes a dissertation that will explore the steps necessary to both build and configure Ultra-Low Power FPGAs. These steps include:

1. revisiting circuit design of FPGA-building blocks and re-designing them for low power operation
2. re-assessing FPGA architectural parameters to determine architectures most suitable for ULP operation for FPGAs as opposed to high-performance operation
3. developing a toolflow that will not only allow researchers to quickly co-optimize FPGA fabrics for different circuit and architectural parameters, but also users in the future to quickly be able to generate FPGA fabrics with the necessary configurations for a given benchmark circuit
4. exploring the integration of the ULP FPGA fabric into a SoC designed for wireless sensing applications

1 Introduction

1.1 Motivation for Low-Power FPGAs - Ubiquitous Computing

Today, in a world where people have immediate access to massive amounts of information, increased control and awareness of the surrounding environment is becoming more and more viable, and perhaps necessary. For this to occur, integrated circuits (ICs) need to be redesigned such that a large number of sensors can be deployed simultaneously in different environments (wearable sensors, smart home sensors, infrastructure monitoring, etc.) and users can access the information provided by these sensors and respond accordingly. This landscape, known popularly as ubiquitous computing (UbiComp), is not far off. There are, however, certain restrictions in this application space that prevent UbiComp adoption on a large, commercial scale. These challenges are as follows:

- **Ultra-Low Power and Energy Operation** – For UbiComp to occur, there will need to be a large number of ICs deployed in various environments (homes, wearable sensors, etc.). Constantly changing batteries for all of these devices, or connecting all of them to wires, becomes invasive and infeasible. Therefore, these sensors need to be very-low power, so as to maintain functionality with a small power budget, and also be low energy, so as to not drain power sources too quickly. This will minimize battery changes, allowing the sensors to spend extended periods of time functioning.
- **Computational Flexibility** – With many sensors deployed in the field, constantly gathering information, removing them from their deployments can be problematic. Moreover, algorithms used for sensing applications are subject to change, either from discoveries of new methods, or from updated versions of the current approach. Additionally, these sensors are subject to different environments, and thus may need to perform slightly (or completely) different functions based on the environments. For all of these reasons, these sensors need to be designed with a level of flexibility, in order to meet all of the requirements of the environments they are sensing.
- **Computational Power** – It's not enough for these sensors just to sense information. If the devices simply sent all information collected to a user, it would be highly inefficient, as the sensor would send more information than is necessary, and the information would likely not be discernible by the user. Ideally, all of the sensors in this UbiComp platform will be able to process some of the information it collects on the node itself, in order to give the information to whoever needs it in a way that is easy to understand and work with.

- **Low Cost** – As mentioned before, UbiComp would require thousands of sensing nodes to be deployed all at the same time. The cost of developing and fabricating these chips would be very high, and that cost goes up even further if these chips need to be re-designed and re-fabricated with each new or updated algorithm.

Unfortunately, current IC options cannot meet all of these requirements. Ultra-low power (ULP) Application Specific Integrated Circuits (ASICs) provide the most efficient computing platform, and can function at low power consumptions with high computing power. Unfortunately, they have little or no flexibility, so updating or changing the functionality of the ASIC would require a full re-design and re-fabrication. This inflexibility also contributes to higher costs for implementation. General purpose processors (GPPs) that are targeted for low power, like ULP microcontrollers (MCUs), like the MSP430 from Texas Instruments, boast low power operation high flexibility and computing power[13]. However, the power consumptions of these devices is still too high for certain applications within the UbiComp space.

A more viable option is a ULP system-on-chip (SoC), like the Body-Area Sensor Node (BASN) described in [1]. This device leverages both MCU and ASICs to get extremely low power consumptions ($<20 \mu\text{W}$), allowing the node to run on harvested energy. Battery-less operation is *perfect* for UbiComp applications. Unfortunately, devices like this one still suffer from lack of flexibility. Although the node can perform many functions (ECG, EEG, aFib detection, etc.), it is limited to those functions, and needs to be re-fabricated and re-designed for any changes or updates.

Field Programmable Gate Arrays (FPGAs) are reconfigurable ICs that, at nominal voltages, split the difference between ASICs and GPPs in terms of flexibility and efficiency. The reconfigurability of the devices makes them inherently more flexible than ASICs, as they can be reconfigured to any functionality that fits on the logical resources available. Because FPGAs are still hardware implementations of algorithms, they still consume less power and energy than GPPs for many applications. The flexibility of FPGAs is limited to the amount of logic resources on the FPGA, which limits their flexibility. Both ASICs and GPPs have been retargeted for ULP operation, as discussed earlier. However, there are fewer options for ULP FPGAs. This is primarily due to the current market for FPGAs. Applications today that primarily use FPGAs include aerospace, defense, automotive, High-power consumer electronics (like digital cameras), and medical devices (ultrasound, endoscopes, etc.), among others[14]. For these applications, power and energy consumption are not as important as performance. Thus, the FPGA industry has been driven by a different set of metrics, and designs for FPGAs have progressed accordingly. That being said, there are FP-

GAs from companies like Microsemi and Lattice Semiconductor that boast extremely low power consumptions[11][12]. But again, those devices still consume 10s of mW in active mode, which is high for the UbiComp requirements. If sub-mW FPGAs were to be realized, they would provide an ideal solution for UbiComp applications; a power-efficient computing platform that is also flexible and powerful. This proposal argues that if the circuits and architectures of FPGAs are retargeted for ULP operation, FPGAs can fill the space needed to realize UbiComp pervasively.

1.2 Motivation for Configuration Scheme for Custom-FPGAs

In order to adequately test whether circuit and architecture optimizations are valid for minimizing power consumption in FPGAs, it will be important to test how incremental changes effect the FPGA circuit as a whole. To do this, we will need to build schematics of FPGAs with the different circuit-level and architectural parameters, and configure them to perform different functions. This poses two major problems.

- **Design Time** FPGAs are large structures, which require the use of a large number of transistors. One single configurable logic block (CLBs) consisting of 9 4-input basic logic elements (BLEs) can have as many as 20,000+ transistors (including test structures like registers). Thus, building entire FPGAs, which include multiple CLBs and a global interconnect, is very time consuming. Multiply that by the different FPGAs required for a thorough design space exploration, and comparing FPGA parameters through SPICE-level simulations becomes virtually impossible by hand.
- **Configuration for Custom-built FPGAs** As it stands now, there is no commercially available way to configure a custom-built FPGA fabric. Commercial FPGA companies, like Xilinx and Altera, have their own compilers that works for mapping a configuration bit stream that can be loaded to their hardware to implement verilog code provided by the user. Unfortunately, this method will not work for custom-FPGAs, because the custom hardware (circuit designs and architectures) doesn't match the commercial tools. Configuration bit locations are different, so the configuration bit stream wouldn't configure the same configuration bits. Configuration mechanisms vary from FPGA to FPGA, so it's entirely possible that the configuration bit stream generated by the commercial tool would be completely invalid for the custom FPGA built. Moreover, the underlying architectures will vary for commercial FPGAs, meaning that algorithms will be partitioned to different amounts of logic and interconnect resources. For all of these reasons, using commercial configurations is impossible for custom

fabrics.

What is needed is a method of taking a set of circuit level and architectural parameters (which define your physical FPGA structure) and a function for the FPGA to implement, and generating from those a netlist ready for simulation.

1.3 Thesis Statement

By re-targeting circuit elements and architecture parameters of reconfigurable fabrics for ultra-low power operation, FPGAs can provide the adequate combination of efficiency, flexibility, and computing power to enable ubiquitous computing to become a reality. These fabrics could be deployed as stand-alone ICs, or be included in ultra-low power SoCs to increase their flexibility. In order to study FPGA circuits and architectures fully, it will be necessary to generate FPGA fabrics, and configure them with algorithms used in ULP applications, in order to see how they will function. To do this, open source FPGA mapping tools (i.e. VTR) need to be extended to generate configurations for FPGAs.

1.4 Goals

The overarching goal of this work is to explore and determine circuit designs and architectures to allow FPGA operation with extremely low power consumption ($<1\text{mW}$). The individual goals of the work are as follows:

- Explore viable circuit designs for FPGA sub-circuits (logic blocks, interconnect structures, and configuration bits)
- Determine best practices for FPGA sub-circuit design
- Re-visit FPGA architectures to determine if there are better architectural parameters for ULP FPGAs
- Determine optimal FPGA architectures for different classes of ULP applications
- Develop a tool-flow enabling circuit-level and architectural co-optimization for FPGA fabrics
- Leverage toolflow to create FPGA fabrics that perform tasks with ultra-low power and energy consumptions
- Use toolflow to help create embedded reconfigurable fabric to be used in ULP SoCs, illustrating the benefits that ULP FPGA fabrics can provide

2 Background

2.1 General FPGA Architecture

FPGAs are reconfigurable ICs that consist of a collection of distributed logic blocks whose inputs and outputs are connected to each other through a reconfigurable interconnect, filled with switches and buffers. Configuration bits (usually SRAM) are distributed through the FPGA and control the reconfigurable interconnect and the logic, and these configuration bits, if altered, can change the functionality of the logic blocks, the connections between different logic blocks, or both. The I/Os to the FPGA chip also include configuration bits that determine whether it will be an input or an output, allowing flexibility in the number of input and output pins the device has. A high-level illustration of the general FPGA architecture is shown in Figure 2.1.

Each configurable logic block (CLB) consists of one or more Basic Logic Elements (BLEs). These

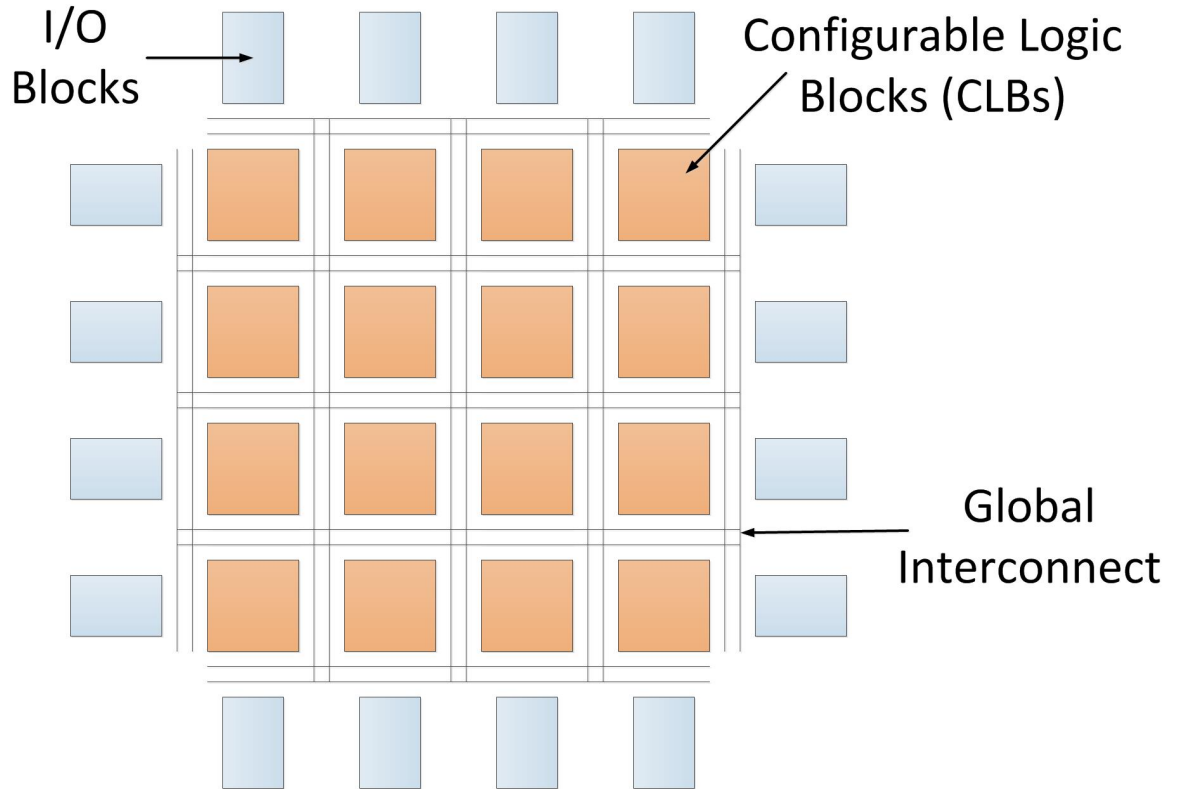


Figure 2.1: A 4x4 2D-mesh FPGA. Configurable Logic Blocks (CLBs) are distributed and connected to each other and input/output (I/O) blocks by a reprogrammable global interconnect.

BLEs include a look-up table (LUT), a flip-flop (FF) and a multiplexer (mux). A k -input LUT is implemented by 2^k SRAM configuration bits that hold the truth table values of any k -input Boolean function. Those bits are then fed into a 2^k -to-1 mux, whose select signals are controlled

by the inputs of the BLE. The output of the LUT is then connected to both an additional output mux and the input of a FF, whose output is connected to the other input of the output mux. The output mux is controlled by a configuration bit, which determines if the BLE is purely combinational, or if it will utilize the FF to perform sequential logic. A detailed image of the BLE is shown in figure 2.2. The global FPGA interconnect is generally organized as a 2-dimensional mesh structure

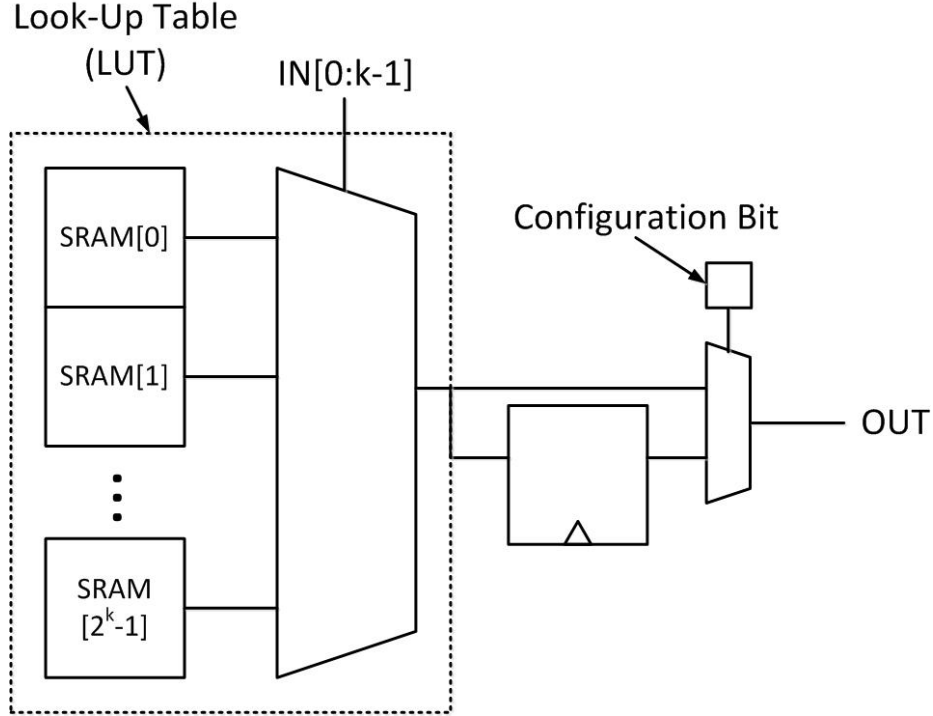


Figure 2.2: A basic logic element (BLE). One k -input look-up table (LUT) has 2^k SRAM configuration bits that are connected to a 2^k -to-1 multiplexer. The LUT is either configured to be combinational or sequential by a multiplexer and a configuration bit at the output.

with horizontal and vertical wires. In 2D-mesh FPGAs, each individual wire is considered a "track," and the collection of adjacent tracks in a given direction are called "channels." Horizontal and vertical channels are connected to each other through switch boxes. Connection boxes connect the CLBs to the FPGA interconnect wires. Configuration bits control which tracks are connected to each other and to the CLB inputs and outputs. Figure 2.3 gives a detailed picture of interconnect resources and how they are connected.

2.2 Commercial Ultra-Low Power FPGAs

Most of the market share for FPGAs is shared by Xilinx (47%) and Altera (41%) as of 2012[17]. However, these companies do not make FPGAs with low power consumptions. These companies'

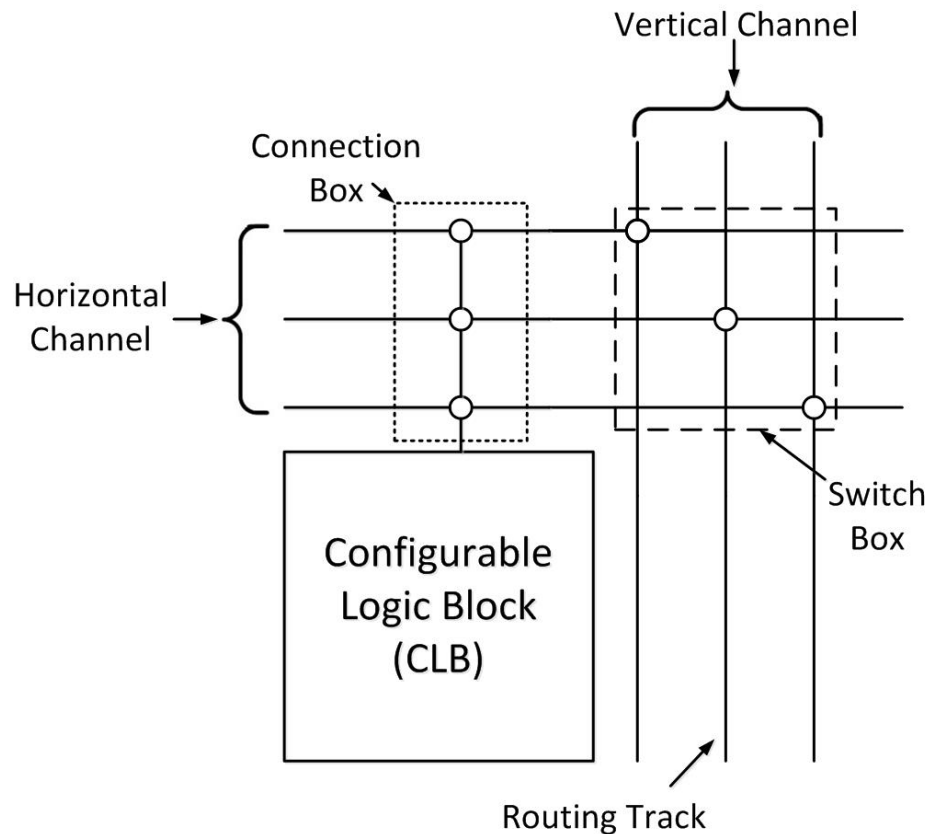


Figure 2.3: Programmable FPGA interconnect. Connection boxes connect signals from CLBs to individual routing tracks of horizontal and vertical routing channels. Switch boxes connect the channels to each other.

current consumption are consistently in the mW-W range, which is too high for ULP applications. There are companies, however, who do make FPGAs with power consumptions in certain modes that are sub-mW, and their circuit designs and architectures are described in this section.

2.2.1 Microsemi IGLOOnano

One of the competitors to Xilinx and Altera is Microsemi, a company that makes some of the lower power FPGAs on the market. Their devices use non-volatile FLASH configuration bits instead of the SRAM bits that most industry FPGAs use. By using these, they can get leakage currents for their devices as low as $2 \mu\text{W}$. The smallest logical element in the IGLOO nano, which is analogous to a BLE, is called a VersaTile, which is a 4-input block that can have one of three functions:

1. 3-input LUT
2. Latch with clear or set

3. D flip-flop with clear or set

The IGLOO has no explicit clustering of their VersaTiles, but clusters are implicitly created by the routing structures in the FPGA. There are multiple levels of hierarchy in the interconnect of the IGLOO. First, there are Local-Line resources that connect every VersaTile with its eight nearest neighbors. Next, Long-Line resources span 1, 2, or 4 VersaTiles, in either the x or y direction. Lastly, there are Very-Long-Line resources that act as the global interconnect for the FPGA, which span 12 versatiles in the x-direction and 16 VersaTiles in the y-direction. Because of the Local-Line interconnects, we will consider the cluster in these FPGAs to be 9 VersaTiles[11].

2.2.2 Lattice iCE40 Ultra

Lattice Semiconductor is a company who designs SRAM-based FPGAs for low-power operation. Their iCE40 device boasts typical standby currents of 71 μA , with IP cores including embedded block RAMs, communication interfaces, and DSP blocks. The device clusters 8 4-input LUTs into each programmable logic block (PLB). Like other commercial FPGAs, their logic blocks have dedicated carry logic for more efficient arithmetic functionality. Their routing structure consists of three different segment lengths spanning 1, 4, and 12 PLBs[12].

2.3 Academic Ultra-Low Power FPGAs

While some commercial FPGAs (like the Microsemi IGLOO and the Lattice iCE40) have sub-mW operation, they only achieve this low power consumption in sleep mode. There have been academic ventures to design ultra-low power FPGA designs that consume low power actively. One example is introduced in [7], which features a low power FPGA that consumes as little as 40 μW in the active mode. This FPGA uses 6T latches instead of SRAM bits that are used in commercial FPGAs. For interconnect resources (such as connection boxes and switch boxes), this FPGA uses buffered, unidirectional wires with buffered multiplexers at the switch points. Limitations to this FPGA design come mostly from the size of the device. This FPGA achieves record-low power consumption, but only has 4 configurable logic blocks. That severely limits the computation power of the device, and almost assuredly is not enough resources for many of the DSP algorithms needed for the IoT.

Another low-power FPGA, proposed in [8], uses the Xilinx Stratix III architecture as its baseline. Thus, it also uses standard SRAM configuration bitcells and buffered unidirectional multiplexers for the interconnect routing as well. This FPGA included IP blocks like multipliers and block RAMs. The low-power optimizations employed in this FPGA include mid-oxide, high-VT transistors in the

Table 1: FPGA architectures of commercial and academic low-power FPGAs

FPGA	Logic element inputs (K)	Logic element modes	Clustering (N)	Segment Lengths (L)	Configuration Bit Type
Lattice iCE40	4	– 4-input LUT	8	4, 8, and 12	SRAM
Microsemi IGLOO nano	4	– 4-input LUT – Two 3-input LUTs (w/ shared inputs)	9	1, 2, 4, 12, and 16*	Flash
Ryan et. al [6]	4	– 4-input LUT	9	1	5T-SRAM
Grossman et. al [7]	4	– 4-input LUT	8	1	6T Latch
Tuan et. al[8]	4	– 4-input LUT	8	1	SRAM

* – These segments span one VersaTile (BLE) not a CLB

configuration bits. They also power-gate unused resources, and have a stand-by sleep mode. With these techniques, this FPGA consumes 13-130mW of active power and 46-460 W of sleep power for 1500-15000 logic cells (BLEs).

In [6], researchers also used SRAM bitcells for configuration, but used 5T cells as opposed to the standard 6T. This made the cells harder to write, but because the cells are generally in the hold state, using 5T configuration bits only requires extra effort during the configuration stage, in the form of boosted voltages for the writing circuitry. For the interconnect, this FPGA doesn't use buffers at all, but instead uses a purely pass-gate interconnect. To make up for the degraded swing across the interconnect, a modified-Schmitt Trigger sense amp is used at the inputs to the logic blocks in order to restore the signal to full swing. This interconnect leverages the low-swing that occurs throughout the interconnect to reduce power. Using these circuit design techniques, researchers were able to reduce area per LUT by almost 3x, delay at a constant energy by 14x, and energy at a constant delay by almost 5x.

Table 1 shows different low-power FPGAs and their architectures. Although other commercial FPGA companies (like Xilinx and Altera) have moved to larger LUTs, low power FPGAs have interestingly stayed with 4-input LUTs, and all have similar clusterings. Segment lengths and configuration bit topologies, however, vary among the FPGAs. It's important to note that the segment lengths for the IGLOO nano are with respect to their logic elements, called VersaTiles, and not to their clusters of VersaTiles.

3 Circuit Exploration

3.1 Motivation

FPGAs are often overlooked for ULP applications primarily because of the inefficient overhead incurred from reprogrammability. Logic is represented through Look-Up Tables (LUTs), which implemented as memory cells connected to a set of multiplexers. This implementation is inherently more inefficient than an ASIC implementation of the same logic. Moreover, FPGAs have reconfigurable interconnect that connect the different logic blocks together, resulting in unused routing resources and configuration memory cells used to select which routing resources are active. ASICs, on the other hand, only have the routing necessary to perform the given function, which greatly reduces the amount of circuitry outside of the logic path. The first step to lowering the overall power and energy consumption of FPGA fabrics is to optimize each FPGA sub-circuit. FPGAs have relatively few sub-circuit building blocks, making an exploration of each of them feasible. These blocks include interconnect resources (switch boxes and connection boxes), configurable logic blocks (CLBs), and the configuration bits used to store configuration information for the FPGA. For FPGAs to be feasible for ULP applications, their energy consumption must be comparable to that of ASICs. To achieve this, each individual circuit element must be re-optimized for new metrics, namely low energy and small area. Once each of the circuit elements are redesigned for more efficient operation, the overall power consumption of the FPGA will be reduced dramatically, closing the gap in efficiency between FPGAs and ASICs. The challenge here is to find new designs for these building blocks that have better power and energy consumptions than the state-of-the-art FPGA sub-circuits, while maintaining a level of flexibility suitable for UbiComp and other ULP applications.

3.2 Prior Art

In[9], researchers compared different FPGA interconnect switches to determine which would reduce power consumption most effectively. In this case, they compare the traditional routing switch, which they define to be a multiplexer followed by a buffer, with different low-power variants, which include adding headers to provide different modes (high speed, low-power, and sleep mode). There is also an additional buffer topology, which takes advantage of body-biasing the PMOS devices in the buffer to the virtual rail that is provided by the header devices in the switches. Looking at all the different combinations of techniques, they find that using any of the routing switches lowers dynamic power over the traditional switch topology about 30%, low power modes lower leakage

power 28-52%, and sleep mode lowers leakage 61-79%. This work can be extended by looking at un-buffered interconnects, and also by attempting to incorporate a study of the effects of variation. CLB topologies have not been investigated thoroughly. [6] introduced an alternative idea from the standard multiplexer-based local interconnect inside CLBs which uses FPGA-style routing to connect different BLEs within a CLB together. [10] introduced the notion of depopulating, or removing full connectivity in intra-CLB routing between the inputs/outputs of the CLB and the individual multiplexers inside. However, to the knowledge of the researchers involved, no study has yet been conducted comparing CLB connectivity topologies against each other across a range of circuit-level knobs. [7] compared different configuration bit topologies, including standard 6T SRAM, subthreshold 6T SRAM, and a 6T latch. His study found that the 6T latch had the most robust propagation delay, but the slowest write time. The study correctly pointed out that write pulse width is not an important metric for discussing configuration bit topologies. To extend this study, looking at factors such as hold margin, minimum retention voltage, and leakage current should be highlighted for different configuration bit topologies. [8] compared the differences between using simply high-VT devices vs. using mid-oxide devices, with higher dielectric constants than the other high-VT devices. They found that using transistors that are both mid-oxide and high-VT reduces leakage power by multiple orders of magnitude. In this work, transistor type as well as configuration bit topology will be knobs to turn for optimizing configuration bits for low-power operation.

3.3 Research Question

How can we redesign the circuit elements in FPGAs (namely routing switches, CLBs, and configuration bits) to minimize power consumption, while still providing adequate functionality and performance for low-throughput applications?

3.4 Approach

The overall goal of this portion of the research will be to search for better alternatives to the current state-of-the-art FPGA sub-circuits. This research will focus primarily on the topologies of switch boxes, CLBs, configuration bits, and LUTs. Other sub-circuits, such as connection boxes, I/Os, registers (within the CLBs) will not be discussed in this work, and are left for future work.

3.4.1 Routing Switch Exploration

Different routing switch topologies will be compared to find the optimal low-voltage (near- to sub-threshold) option. We compare the different topologies using an interconnect model, which consists of a driver, a series of switch boxes, and one or more receiver circuits. This model is a preliminary model, and the results from analyzing this model will be presented in this proposal. In the future, a custom-built toolflow (which will be discussed later in this document) will be able to generate full FPGA schematics, and thus test how changes in switch topology effect the FPGA as a whole. This will allow for comparisons between the different switch box topologies across all sorts of benchmarks, creating a more comprehensive exploration of the switch-box-topology design space. In this research thrust, we will explore using tri-state buffers (TSBs), pure pass gate (PG), pure transmission gate (TX), and multiplexer-based designs for the switches in the switch boxes. We will first compare the individual switches, followed by their functionality through specified networks. We will also explore both bi-directionality and uni-directionality in the context of ULP operation. The knobs we will turn for this comparison are:

- **Transistor Size** – Increasing transistor size for the routing switches increases driveability and performance, but also increase load capacitance and area, and thus power consumption. The effect of transistor size on interconnect metrics is complicated, and needs to be studied.
- **Path Length** – Interconnect nets in FPGAs vary greatly in length, with nets ranging from one wire segment long to the diagonal of the entire chip. It will be important to discover and quantify how different routing switch topologies change their performance and power consumption across different path lengths.
- **Operating Voltage** – Power consumption has a quadratic relationship to operating voltage, so operating the interconnect at the lowest possible voltage is optimal. However, it is expected that the power consumption of different design topologies can have varying changes in metrics like power consumption and performance across different operating voltages.
- **Transistor Type** – Different types of transistors, with varying V_T 's and dopings are available to be used. Leveraging the ability to use these different types of transistors could result in increased power and energy savings.

3.4.2 Configurable Logic Block (CLB) Exploration

Configurable Logic Blocks (CLBs) are the building blocks of FPGAs, and house all of the logic performed by them. Thus, optimizing the CLBs for ULP operation will be paramount for developing ULP FPGAs. In modern FPGAs, CLBs are collections, or clusters, of basic logic elements (BLEs). The focus of this exploration isn't to change the topology of the BLEs themselves, but rather redesign the local interconnect between the BLEs inside the CLB. Modern FPGAs (for the most part) currently use multiplexers to route the inputs and outputs of the CLB with the individual outputs of the BLEs on the inside. However, [6] proposed an alternative solution, building a mini-FPGA style CLB, which uses switch boxes and connection boxes to connect the BLEs. In that design, researchers were able to achieve a smaller CLB area and lower power. Figure 3.1 illustrates the two topologies. In this work, we will further explore the design space, both in simulation and in hardware, and see under which set of parameters each of the two CLB topologies is optimal. The

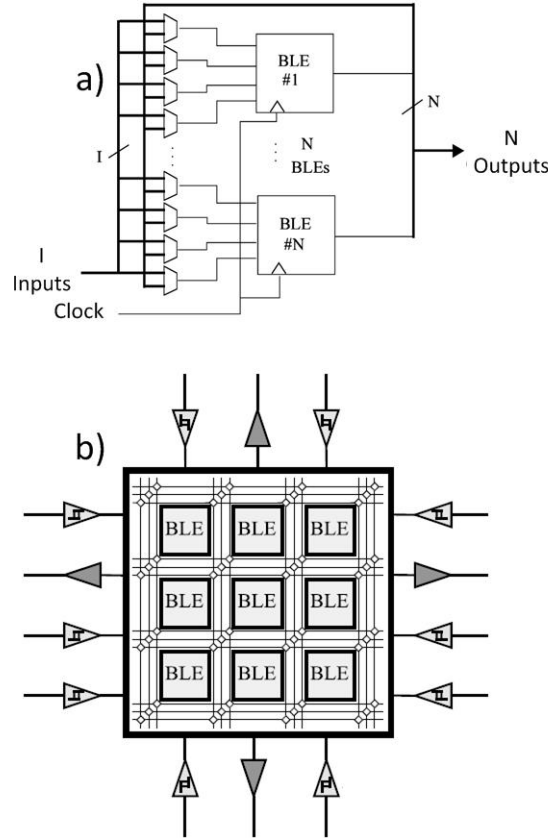


Figure 3.1: a) multiplexer-based[?] and b) mini-FPGA[6] Configurable Logic Blocks (CLBs)

mux-based CLB has large muxes at each input of each BLE that allow for connectivity to any of the inputs or outputs of the CLB. In the mini-FPGA case, BLEs are treated like CLBs in an FPGA,

connected by a network of switch boxes, connection boxes, and routing wires. The knobs we decide to turn for this CLB exploration are:

- **K** – Number of inputs to the individual BLEs of the CLB
- **N** – Amount of clustering, or the number of BLEs total in the CLB
- **Depopulation** – This is a measure of how much the size of the input multiplexers in the mux-based CLBs is reduced, measured in percentage (for example, 50% depopulation would allow connectivity to half of the signals). This is a common technique used in FPGAs to minimize the multiplexer size in the CLBs, reducing area and power consumption.
- **Channel Width** – This parameter refers to the number of routing channels are present in the local interconnect of the mini-FPGA CLB. Reducing the number of channels helps lower power consumption and area, but limits routeability through the CLB.
- **Operating Voltage**

3.4.3 Configuration Bit Exploration

Configuration bits are large contributors to the overhead of FPGAs over ASIC designs. These bits control the data inside each of the logic blocks, combinational vs. sequential modes for logic blocks, and the large global interconnect. Commercial FPGAs can have on the order of millions of these bits. Because of this, it is important to optimize these circuit elements. To do so, we will compare different memory cell topologies in a way that pertains directly to FPGA functionality. Generally, when looking at different bitcell topologies, researchers tend to focus on reading the data from the memory cell and writing data into it, making sure to balance both of the phases while also minimizing energy consumption. In FPGAs, however, the read and write phase are not important while the device is in use; only the hold phase is. Thus, this configuration bit exploration will be similar to those that have been discussed in the past, but with more of an emphasis on hold margin and leakage reduction. Our approach will be to take multiple bitcell topologies, including but not limited to:

- Standard 6T SRAM
- 5T SRAM (Assymetrical, proposed in [Ryan])
- 6T sub- V_T SRAM

- Sub- V_T 8T SRAM
- 6T sub- V_T latch (proposed in [Grossmann])
- SRAM w/ low-power header (presented in [Anderson])

These bitcells will be simulated using SPICE with few different tests, both static and dynamic. We will vary transistor sizes for each of the bitcell topologies (or use existing knowledge) to determine the optimal sizing for the each bitcell topology we test. If given the opportunity, we will also tapeout banks of the different bitcell topologies to gather hardware test results as well, and compare them to simulation.

3.5 Evaluation Metrics

In our study of routing switches to be used in the FPGA interconnect, our metrics for evaluation include:

- **Area** – The switch matrices in an FPGA account for the largest percentage of the area in an FPGA. Reducing the area of the switch box, then, greatly decreases the area of the whole FPGA, thus also decreasing power and energy consumption. Area will be estimated by transistor count.
- **Power Consumption** – In trying to find an optimal design for ULP FPGA design, it will be important for the block to consume the least amount of power possible, to stay within the strict power constraints of ULP applications. We will measure power consumption through simulations of the device performing our given task.
- **Energy Consumption** – While it is clearly important to minimize power for devices in ULP applications, it is often the case that ULP applications have strict energy consumption constraints as well, particularly if the device runs on a small battery or harvested energy. Thus, while timing constraints might be lax, energy consumption still needs to be low to extend battery life.
- **Robustness** – Maintaining proper functionality in the presence of process-voltage-temperature (PVT) variations is especially important when discussing ULP applications, because the physical parameters of the devices are even more sensitive to these variations.

The evaluation metrics for the CLB comparison are:

- **Area** – We will attempt to reduce CLB area as much as possible. Smaller CLB area corresponds to smaller capacitances, corresponding to lower power dissipation and energy consumption inside the CLBs. Moreover, smaller CLBs also reduce the wirelengths of both the intra-CLB and global interconnects, and therefore the power dissipation and energy consumption.
- **Power Consumption**
- **Energy Consumption**
- **Routeability** – More depopulation (for mux-based CLBs) and fewer routing channels (for mini-FPGA CLBs) reduce power, energy, and area, but also reduce the flexibility and potential functionality of the CLB. We want to have a metric for how much that flexibility to realize different circuits is reduced. There are many ways we could calculate, but we will simply represent routeability as the number of possible inputs to a pin.
- **Robustness** – measures of robustness for the purposes of this study will be more for individual circuit elements (muxes vs. routing resources) then they will be for the full CLBs. The CLBs are so large (can be 1000s of transistors) that Monte Carlo simulations, which would be conducted in order to characterize circuits in the presence of variation, would not be viable. The evaluation metrics for the configuration bitcell topologies are:
 - **Leakage Power** – Along with hold margin, this is one of the more important metrics by which to evaluate the different configuration bit topologies. These configuration bits are some of the most common circuit elements in an FPGA, so limiting the leakage power of these bitcells will have a strong impact on the overall leakage power of the device.
 - **Hold Margin** – Another important attribute for the configuration bitcells. For FPGAs, the most important function of a configuration bitcell is to retain its state. Thus, it will be important to quantify how much voltage swing on a bitline a configuration bit can tolerate without losing its value while access to the bit is denied.
 - **Retention Voltage** – Reducing voltage is the most effective way of lowering power consumption. As a result, it will be important to characterize different bitcell topologies based on how low the supply voltage of the bitcell can be reduced and still reliably retain the information stored inside.

- **Write Voltage Boost** – A lesser important metric. When writing some of these bitcells that are designed specifically to hold without consideration given to read- and write-speed, it should still be understood what kind of overheads will be included in being able to successfully write to the cell.
- **WL Pulse Width** – This metric quantifies the time it takes to write into each individual cell. This metric is again less important, because writing to the bitcell happens very rarely in FPGA lifetime, compared to regular functionality.

3.6 Results

3.6.1 Routing Switch Exploration

The work presented in this proposal extends the work done in [6]. That FPGA design used pass transistors as interconnect routing switches, and proposed a modified Schmitt trigger sense amp receiver that transitions low-to-high at a lower threshold voltage than the traditional buffer, thus drastically lowering the static current that results from a low-swing input. In this work, we compare that design to other low-swing sense-amp designs to see if there are better solutions. Figure 3.2 shows circuit topologies of different sense amp topologies. The double-buffer is a low swing and a high swing buffer multiplexed together, so that depending on the transition, the proper buffer is used to maximize speed and limit static current. The low-swing buffer is very similar to the modified Schmitt trigger, only with the hysteresis transistor removed. Figure 3.3 shows the voltage

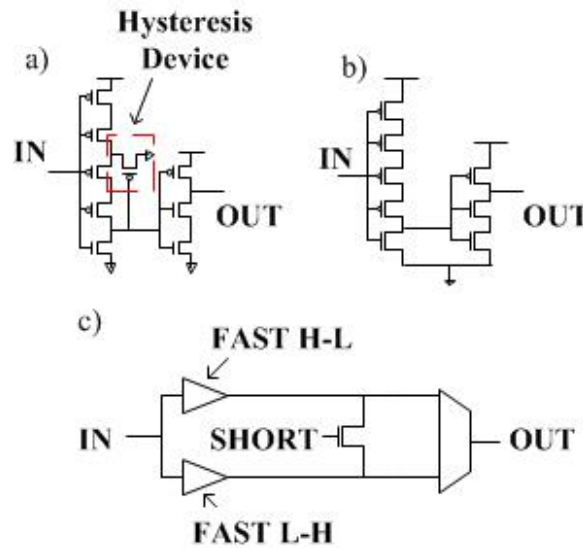


Figure 3.2: a) Modified Schmitt trigger, b) low-swing buffer, and c) double buffer sense amp topologies.

transfer characteristics (VTCs) of the different sense amp topologies, as well as the static current consumption across input voltage. In addition to the three custom receivers, we also compare a buffer built from two inverters and a conventional Schmitt trigger. In this plot, the sense amps are compared at an operating voltage of 0.3 V. The double buffer has the best VTC overall, because it has the lowest low-to-high transition, but also the highest high-low transition. This means the device will switch quickly in both transitions, and should have decently low static current. It is also important to note that for both Schmitt trigger designs, the high-low transition is at a lower voltage than the low-high transition, which results in very slow high-low transitions. The low-swing buffer has the smallest static current, which also peaks at a lower voltage. As a result, high-low propagation delay is higher for Schmitt triggers, and the static current in these transitions is much higher. In

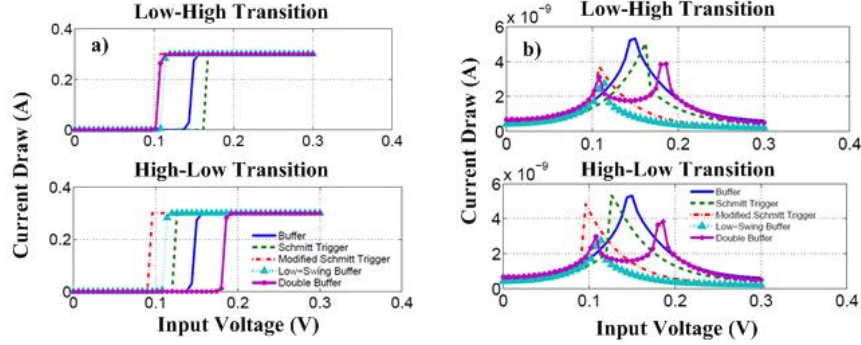


Figure 3.3: a) VTCs and b) static current profiles for the low-high and high-low transitions of the different sense amp topologies at $V_{DD} = 0.3$ V. The double buffer has the fastest low-high and high-low transitions, and the low-swing buffer has the smallest static current.

figure 3.4, we compare the energy-delay (ED) curves of the different sense amp topologies. The low swing buffer is generally faster and lower energy than the other alternatives. Each of the sense amps were simulated with 50 passgates leading to them, in order to accurately depict low-swing inputs. In this plot, voltage was swept from 0.3–0.8 V. The low swing buffer is the optimal solution, minimizing both energy and delay in almost all cases. Only with the double buffer at very low voltage (0.3 V) can you achieve a slightly faster delay for the same energy consumption. Now that we have a good receiver circuit, we can compare the interconnects again. Figure 3.5 shows the ED curves of the tri-state switch box compared to the pass gate switch box. With the low-swing sense amp, the pass gate interconnect design shows lower energy consumption across VDDs. With the same delay (of about 10 ns), the pass gate network alternative decreases energy consumption by 87%. In deep subthreshold ($V_{DD} = 0.25$ V), we find that the pass gate network is 63% faster as well.

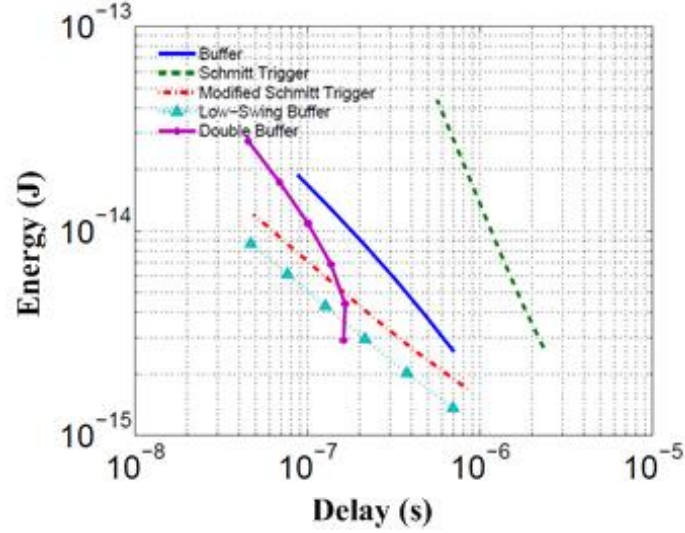


Figure 3.4: ED Curves for sense amp circuits, driven by 50 pass gate switch boxes in series. The low-swing buffer is the overall best option for ULP design

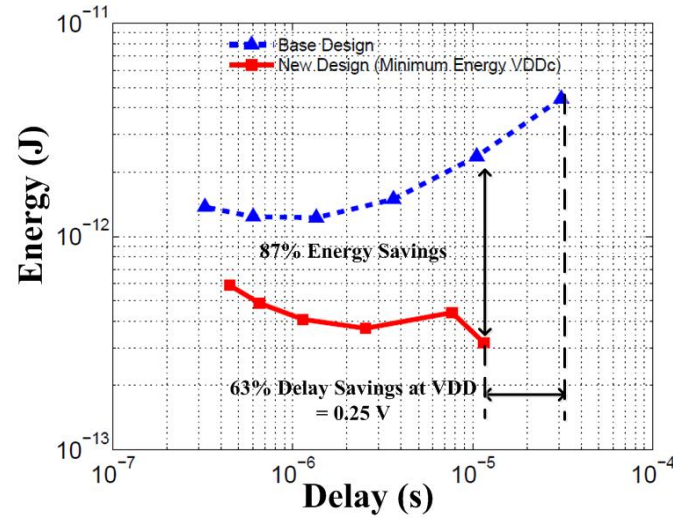


Figure 3.5: ED curves for interconnect networks, sweeping operating voltage. The pass gate interconnect topology is lower energy across operating voltages, and is faster at lower operating voltages.

3.6.2 CLB Exploration

Figure 3.6 plots the total number of transistors attributed to the local interconnects of both multiplexer-based and mini-FPGA CLBs against clustering. The transistor counts for both fully buffered and unbuffered ($T_{buf} = 0$) multiplexers are plotted to show the bounds of possible transistor counts for multiplexer-based CLBs. The number of transistors in the mini-FPGA style CLBs increases linearly with the size of the cluster, whereas the count for the mux-based CLB increases exponentially. The buffered mux-based CLB has more transistors than mini-FPGA CLBs with a channel width of 8

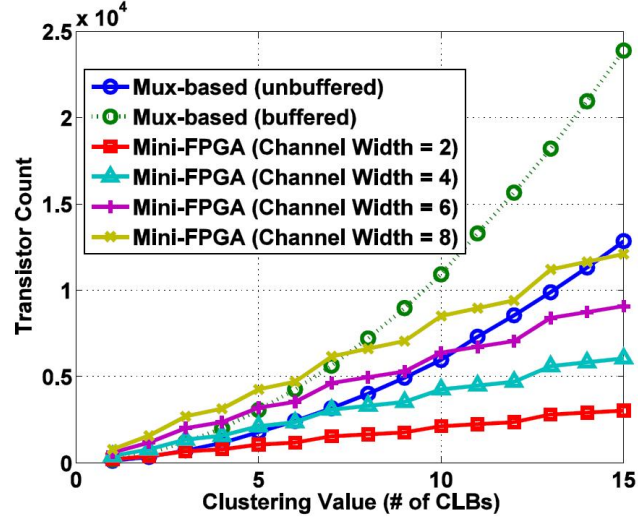


Figure 3.6: Total number of transistors vs. clustering value for different CLB types. Multiplexer-based CLBs become very large with high clustering values. The channel width of Mini-FPGAs contributes heavily to the transistor count.

at a clustering factor of 9, which is smaller than the best clustering value for CLBs as determined by [2] and [3]. It has been empirically determined that a channel width of 6 for the mini-FPGA CLB is sufficient for mapping the entirety of the MCNC Golden 20 benchmarks, which are often used to benchmark FPGAs. At this channel width, the transistor count of the mini-FPGA CLB with 15 BLEs (9072 transistors) is approximately 38% of the buffered multiplexer-based CLB (23880 transistors), and 87% of the unbuffered multiplexer-based CLB (12840 transistors).

Figure 3.7 is a table which shows the cluster sizes at which the multiplexer-based design and the mini-FPGA design have approximately the same transistor count, which will be referred to as the break-even point. At cluster sizes higher than this break-even point, there is a smaller area for mini-FPGA style CLBs versus the multiplexer-based designs. As the channel width increases, the break-even point happens at larger cluster sizes, as number of switch boxes and connection boxes increases. For a channel width of 6, the transistor count can break even at around 5 BLEs per CLB, if the multiplexer is fully buffered. By making the dependency of transistor count linear (in the mini-FPGA design) instead of exponential (in the multiplexer case), it becomes more feasible to further increase the cluster size of CLBs.

While discussion of transistor area is important, it is also important to characterize the effect of CLB implementation strategies on the overall functionality of the FPGA as a whole. For the purposes of this comparison, we will look at delay for a notion of CLB performance, and energy as a metric for CLB efficiency. We chose architectural parameters for the simulation that were consistent with

Mini-FPGA vs. Mux (Buffered) - Break Even Points				
	K = 4			
Channel Width	Break Even Points @ Different Depopulation %'s			
	0%	50%	66%	75%
2	Always Less	N = 4	N = 5	N = 6
4	N = 3	N = 8	N = 11	N = 14
6	N = 6	N = 11	N = 16	N = 22
8	N = 9	N = 15	N = 23	N = 29
	K = 6			
2	Always Less	N = 2	N = 3	N = 4
4	N = 2	N = 4	N = 6	N = 8
6	N = 4	N = 6	N = 9	N = 12
8	N = 4	N = 9	N = 14	N = 16

Figure 3.7: Total number of transistors vs. clustering value for different CLB types. Multiplexer-based CLBs become very large with high clustering values. The channel width of Mini-FPGAs contributes heavily to the transistor count.

the mini-FPGA CLB used in [6]. The LUTs were not given true functions to perform, such as AND gates, for example. Instead, we configured with chessboard patterns, such that the odd bit values were set to 1 and the even bit values were set to 0. This allowed us to test each possible delay path through the CLB by driving the inputs with a binary counter (from 0-15). The delay measurements are pessimistic, as all 9 outputs are observed to find the worst-case delay for the CLB. Using all 9 BLEs will also give the worst-case energy estimate, because circuit elements in all nine of the BLEs will be switching.

Figure 3.8 shows the delay for the two different CLB topologies across supply voltage (VDD), which is swept from sub-threshold (0.3 V) to super-threshold (0.8 V). The two topologies have very similar delays, differing by approximately 7.5% at 0.8 V and by 14.6% at 0.3 V. The mini-FPGA CLB, however, has the better performance, though slightly, across a range of supply voltages, including both sub- and super-threshold. We also observe that delay savings from using the mini-FPGA CLB increase as circuit operation moves deeper into the sub-threshold regime.

Figure 3.9 plots energy per operation against supply voltage for the different CLB topologies. For this comparison, we use a frequency of 10 kHz, giving enough time for the operation of the

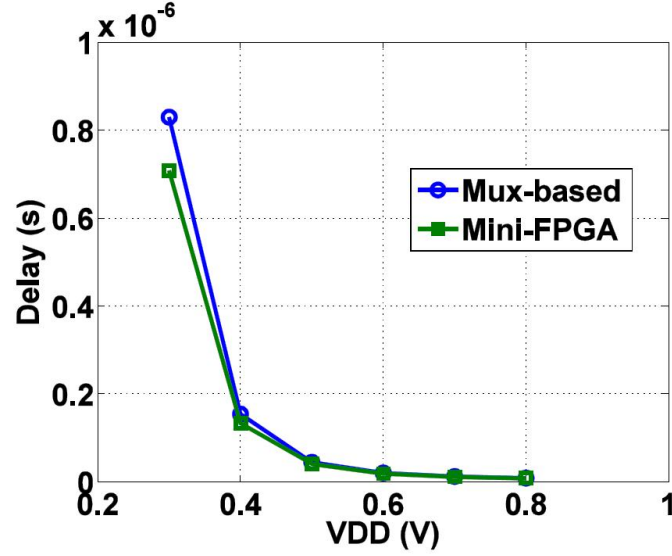


Figure 3.8: Delay vs. Supply voltage for the two CLB types. The two designs have essentially the same delay, w/ the mini-FPGA style having a slight advantage in sub-threshold.

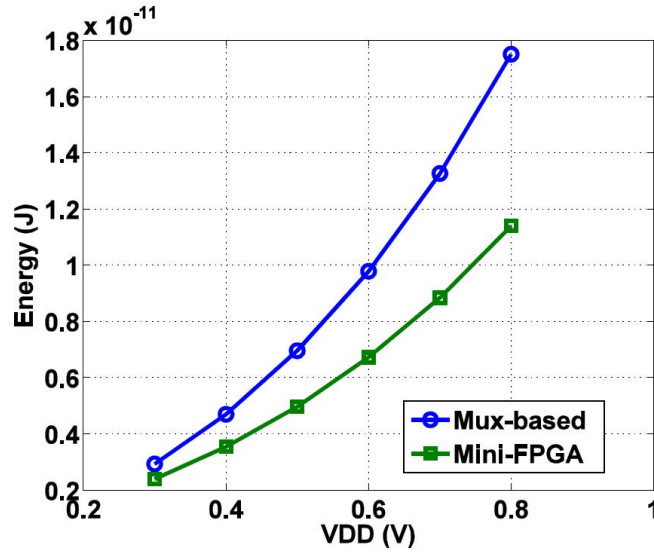


Figure 3.9: Energy vs. Supply voltage for the two CLB types. Mini-FPGA CLB have lower energy across supply voltage than the multiplexer-based CLB due to a lack of switching buffers in the switch boxes and connection boxes.

CLBs at 0.3 V. From this plot, we see that the mini-FPGA CLB is a lower energy option across supply voltages, with an 18.1% reduction at 0.3 V and a 34.9% reduction at 0.8 V. This happens for two reasons. First, each of the 29-input multiplexers in the multiplexer-based CLB has 28 buffers, each which actively draw current as the inputs to those multiplexers transition. This is not true for the interconnects of the mini-FPGA style CLBs, which have no buffers at all in the interconnect.

Second, the mini-FPGA CLB is slightly faster, which will slightly lower the energy due to leakage per operation as well.

3.7 Anticipated Results

There are other switch box topologies that are used by FPGAs, both commercially and academically. Including those in our study of switch boxes will help to make this work more complete. Also, conducting a more thorough design space exploration, in which paths of different depths and branching breadth are also explored, we can understand how different switch topologies work in FPGAs clearly.

For the CLB exploration, the next step is to expand the simulation exploration to encompass more of the design space. The example simulation result is only one small point on that design space. If the same simulation was run with $N = 3$ and a mux-based depopulation of 75%, or at $N = 30$ with a mini-FPGA channel width of 2, the results would be very different, and very contrary to each other. It will be important to look across all of the values around the break-even points in area, and see where the boundaries are for choosing between the two topologies based on performance and energy efficiency, not just in area (figure 3.7). For exploring configuration bit topologies, we will need to characterize different bitcell options, and to compare them based on area, hold stability, and leakage power, both in simulation and in hardware. Because these circuits are targeted to operate at low voltages, variability becomes an even larger issue. Thus, it is paramount to include robustness to variability as a metric, and design our devices accordingly. We will be able to do this by performing MonteCarlo simulations for different FPGA sub-circuits to determine how they operate in the presence of local and global variations.

3.8 Contributions

The contributions of this section are as follows:

- Survey of different techniques for design of FPGA sub-circuits for ultra-low power operation
 - Configuration Bits
 - Routing switches
 - Configurable Logic Blocks (CLBs)
- Design space exploration across different knobs
 - Configuration Bits

- * Circuit topology
 - * Transistor sizing
- CLBs
 - * BLE inputs (K)
 - * Clustering (N)
 - * Multiplexer depopulation (for mux-based CLBs)
 - * Channel width (for mini-FPGA CLBs)
- Routing switches
 - * Circuit topology
 - * Transistor sizing
- Recommendations for circuit-level optimizations for ultra-low power FPGA design

4 Architecture Exploration

4.1 Motivation

In the last section, we discussed how today's FPGAs use circuit elements that allow for increased performance, as FPGA companies design their devices in order to keep up with commercial processors. This is probably even truer of FPGA architectures. The newest generations of Xilinx and Altera chips both have much more complicated CLB structures than the academic FPGA idea, mostly to push performance. In addition, these FPGAs include many different IP blocks for specific functionalities, embedded block rams, and other different IPs. Many FPGAs have extremely large interconnects, because what's more important for some applications is to provide the logic required, and to finish computations as fast as possible (in the GHz range), as opposed to doing so energy efficiently. The problem with these devices is that they are not feasible for ULP applications. Many of these applications require active power dissipation at less than 1 mW. High end commercial FPGAs have sleep power consumption that are still over 1 W. FPGAs that champion low energy operation, like Microsemi IGLOO, still has operating currents in the 10s of mW, which is much lower than the rest of the field, but still about an order of magnitude too high for ULP applications. We propose starting from the beginning, and going back to the architectural sweeps done in the past, but this time trying to minimize power consumption and transistor area, with less of an emphasis on FPGA performance. In doing so, we may find that using different architectures, like different sized LUTs and CLB clusters, may actually be better for FPGAs designed for ULP applications.

4.2 Prior Art

Work has been done to optimize individual architectural parameters for FPGAs. In [2], LUT-inputs (K) and cluster size (N) were co-optimized to minimize the area-delay product of a mapped FPGA. In this study, the VPR tool was used, and the conclusion was that $K = 4$ to 6 and $N = 3$ -10 provide the best results for area-delay product. [3] extends that study by including larger cluster sizes (up to 20), and includes different routing architectures, which take into account both different segment lengths and switch topologies (pass gates vs. buffers). Not only that, but this study also explicitly targets power consumption. In this study, researchers found that $K = 4$ minimized power dissipation across different clustering values, and that a cluster size of $N = 12$ minimized both total power of the FPGA and power-delay product. [5] compares uni-directionality and bi-directionality of interconnects in FPGAs. Uni-directional wiring only allow signals to propagate in one direction

through the interconnect, whereas bi-directionality allows signals propagating in both directions. Findings from this study show that bi-directional routing (in the form of tri-state buffers) for low-frequency designs have lower energy consumptions, but for high frequency operation uni-directional routing is lower energy. The work proposed for this dissertation extends this work by combining the architectural knobs listed above, while also adding additional knobs to the design space (which will be explained in detail later in the section 3.4). This dissertation will also extend this work by leveraging the VersaPower extension to VTR, which we believe will provide more accurate estimations for power consumption, and will take into account all knob changes directly. Lastly, using our tool-flow, we will also be able to generate schematics of the architectures we choose and simulate full FPGAs, adding another level of verification to the architecture exploration conclusions.

4.3 Research Question

How does the optimal FPGA architecture change with a different set of primary metrics, namely area and power consumption?

4.4 Approach

We will first do architecture sweeps for FPGA parameters, assuming the conventional design for the different FPGA sub-blocks (BLEs, interconnect resources, I/Os, etc.) We will do this by leveraging the VTR research tool developed by researchers from the University of Toronto. This tool allows for users start with an FPGA fabric (either a given one or the user can create one) and virtually map a given Verilog or BLIF format algorithm (again, there are given ones, but the user may also create one). The virtual mapping consists of a description of the interconnect (describing the resources used), a netlist describing which blocks are connected to each other, a description of the placement of each block relative to the others, and a set of statistics about the mapping, such as resource utilization, number of routing channels, and critical path delay, just to name a few[18]. For this exploration, we will use VTRs wide variety of knobs to do a thorough design space exploration of FPGA architectures on multiple benchmark circuits. We will then verify the results by doing simulations on a few test cases, leveraging a custom tool flow developed to work with VTR to go straight from FPGA architectural and circuit parameters to FPGA simulation and fabrication (explained later in the document). The knobs for this architectural exploration are as follows:

- **K** - number of inputs to LUTs
- **Clustering (N)** - the number of BLEs in each logic block

- **Channel Width (W)** - the number of routing channels in a routing segment. This, however, can also be an output, which will be explained in the following section.
- **Channel Fanout (FC)** - the number of channels in the global interconnect that a logic block or FPGA I/O can connect to
- **Segment Length (L)** – number of CLBs spanned by individual wire segments.
- **Unidirectionally vs. bi-directionality** - whether or not interconnect tracks allow signals to go in both directions

4.5 Evaluation Metrics

There are two classes of metrics that we will use to evaluate the different architectures. First, we have a set of outputs that come directly from VTR. We will find the architectures that minimize each of those individually, and also develop a figure of merit to lump all of these dependent variables together, in order to find the optimal architecture, based on our set of knobs, for each benchmark circuit we test. We expect there to be an over-arching pattern that holds true for most of the benchmarks, but it is definitely possible for different classes of benchmarks to have different optimal architectures. That will be part of what we find in our study. The metrics of importance that are given in the outputs of VTR are:

- **Channel Utilization** - The percentage of routing resources that are used. High channel utilization results in less wasted interconnect circuitry, and is thus optimal.
- **FPGA Size** – (represented as the number of CLBs) The larger the FPGA, the larger the power consumption. If the same algorithm could be realized in a smaller FPGA, then the power consumption will have to be smaller. Therefore, smaller FPGAs are better for ULP FPGAs.
- **Routing Area, Logic Area, and Total Area** - the same argument applies as with number of CLBs; smaller is better. However, logic area and FPGA size change in opposite ways as clustering (N) changes. Seeing how those two metrics relate to each other, and how they affect routing area as well, will be an interesting part of the findings in this study.
- **Power Consumption** - The newest version of VTR includes a tool called VersaPower, which reports power estimates for some of the given architectures for the VTR tool. We will be able to use VersaPower for the given architectures at the very least, but with some time spent

learning how the software works, we could also expand the tool to allow for architectures other than those given by the VTR tool. It will also be interesting to compare VersaPower estimates to actual SPICE simulation.

- **Channel Width** - Channel width can be a metric as well as a knob to turn. One of the options for running VTR is to leave the channel width un-restrained, and then the program will iterate through channel widths until it finds the smallest possible channel width that completely routes the design.

In addition to using VTR metrics, we will also leverage a custom-built tool flow (which will be discussed later in this document) to generate FPGA fabrics that can be simulated at the SPICE level. The simulations will help verify VTR's estimations, as well as provide a test case for the tool-flow to verify its functionality. The main metrics we will be looking for from the generated FPGAs are:

- Leakage power
- Total power
- Area
- Energy per Operation

We will find one or two optimal architectures from the VPR exploration, and compare those to an FPGA designed using conventional FPGA architectures, to see if the study in fact produced a better architecture for FPGAs targeted for ULP applications. The FPGA fabrics will be configured for applications pertaining to ULP sensing, such as DSP algorithms commonly used by ULP sensors.

4.6 Results

The contributions so far come from initial architecture sweeps of the ALU4 benchmark (one of the "golden 20" benchmarks used in FPGA studies in the past). We swept the number of inputs to the LUTs (K) between 4 and 6, clustering values (N) from 1-10, segment lengths (L) from 1-8, and checked both uni-directional and bi-directional routing. This particular design sweep was conducted using the older Virtual Place-and-Route tool (VPR), a precursor to VTR. Figure 4.1 shows a table that gives the architecture parameters that minimized one of the metrics: total area (as estimated by VPR). A line separates the knobs turned from the metrics measured. You will see that some of the initial sweeps suggest architectures similar to the Xilinx Artix-7 FPGAs (6-input LUTs, Clustering

OPTIMAL ARCHITECTURE - Total Area	
Benchmark	alu4
K-Value	6
N-Value	8
Segment Length	3
Channel Fanout - Input	0.10
Channel Fanout - Output	0.10
Directionality	Uni-Directional
Channel Width	38
Logic Area	2.90E+06
Routing Area	9.07E+05
Total Area	3.81E+06
Critical Path	4.88E-09
Channel Utilization	74.6%
Area-Delay Product	1.86E-02

Figure 4.1: Optimal architecture parameters for running VPR to build an FPGA to map the ALU4 benchmark. Data is misleading, however, as configuration block area doesn't change with increased K or N.

of 8 BLEs, unidirectional routing). This architecture sweep, however, needs to be redone for multiple reasons. First, an old version of VPR was used for this initial architecture sweep. VTR 7.0 is now available, but the figure here was generated using VPR 5.0. Certain features of this older version of VPR limit the effectiveness of the tool for this architecture exploration. First, the logic area for a CLB is a fixed, regardless of the amount of clustering. As a result, when we increase the clustering parameter for VPR, the estimated logic block area remains fixed. Also, as clustering increases, the tools estimate for routing area goes down. Thus, the tool reports larger reductions in area due to clustering than would likely occur. Additionally, the main metric we want to address, power, cannot be addressed by VPR 5.0, but is available for VTR 7.0.

4.7 Anticipated Results

With the newest version of VTR, we will be able to explore power consumption as a function of architecture choice. Ideally, VTR will also address some of the issues posed in the first attempt at the architectural sweep. One of the issues is the unrealistic estimate of logic area that VPR

tool gave. As clustering increases, its value should increase, because there are more logic blocks in the CLB. However, increasing the clustering of a CLB, while the same number of LUTs exist, the local interconnect of each CLB should grow, which should count towards the area of the CLBs. Secondly, only a small portion of the design space is looked at here in this initial sweep. To do a good job, it will be necessary to look at more than one simple circuit. We will be looking at the 20 MCNC Circuits, which are the historical standard for testing FPGAs. However, those benchmarks are quickly becoming obsolete. VTR comes with additional Verilog and BLIF circuits that are more functional in nature, such as adders and multipliers, which will also provide context for real-world applications. We also plan to look at benchmark circuits that have already been implemented in SoCs designed for ULP applications, so as to provide context for the kind of overhead we can expect to implement these circuits on FPGAs. It will also be important to do a more in depth sweep of the different parameters we are looking at. Commercial architectures for FPGAs often provide a variety of different segment lengths and distributions of those segment lengths. Thus, it will be important to also explore some subset of different distributions of segment lengths. The newest version of VTR also allows us to employ fracturable LUTs, something that commercial FPGAs also use, where you have fracturable logic elements (FLEs) which can act as either 1 x-input LUT or 2 y-input LUTs with shared inputs. Adding all of these elements to the design space exploration provides an opportunity to find some very interesting trends, and may potentially result in finding a different architecture than is conventionally used in super-threshold, high-performance FPGAs.

4.8 Contributions

The contributions from this section are as follows:

- Thorough design space exploration of FPGA architectures, turning many different knobs
 - Number of BLE inputs (K)
 - Clustering (N)
 - Channel Width (W)
 - Channel Fanout (FC)
 - Segment Length (L)
 - Unidirectionality vs. Bidirectionality
- Recommendations for architecture parameters for ultra-low power FPGAs

- Both CAD-based and simulation-based exploration
- Comparisons of proposed new architectures vs. current commercial and academic FPGA architectures

5 Toolflow

5.1 Motivation

As discussed earlier in this document, there are many circuit and architectural knobs that we are turning in order to design an FPGA that is optimized for ULP functionality. While testing each individual knob (such as circuit topology) can provide interesting information, its also very important to see how these knob changes affect the functionality of the FPGA as a whole. While FPGA modelling (like what is done in VTR) can give some notion of how the full FPGA fabric behaves with different circuit-level and architectural parameters, a more accurate representation of the circuit performance would be to conduct SPICE-level simulations of FPGA fabrics. To do this, it is necessary to build full FPGA schematics, employing the different circuit-level and architectural parameters we are testing, configuring them to perform some function, and simulating them. However, there are many challenges to conducting these FPGA-level simulations. First, we are exploring multiple knobs, creating potentially 1000s of different combinations for FPGA fabrics. Each one of those FPGAs is also a large IC, using 1000s of transistors. As a result, building FPGA schematics by hand becomes impossible, given time and monetary constraints. Secondly, in order to simulate these FPGAs, they need to be configured to perform certain tasks. Commercial FPGAs have their own compilers for configuration, but these software packages are specific to their own FPGA products, and will not work for our custom-built FPGA fabrics. Thus, a tool-flow needs to be built that not only generates FPGA fabrics based on circuit-level and architectural parameters, but that can also create configurations for those FPGA schematics in order to conduct SPICE level simulations.

5.2 Prior Art

The entire design space of FPGA hardware design can be abstracted to 3 axes: circuit parameters, architecture parameters, and benchmark circuit designs. Commercial FPGA companies, like Xilinx or Altera, allow for large flexibility in terms of the types of circuits that can be mapped to them, but are extremely limited otherwise. They provide only one alternative in terms of circuit parameters, and a small set of different architectures, ranging from smaller, low-end devices to large high performance devices. The VTR tool is slightly limited in the types of benchmarks it can map, but has a large range of architectural flexibility, meaning it can handle a large design space in architectural parameters[18]. Our proposed tool-flow will cover a much larger portion of the possible design space, and be able to cover both architectural and circuit-level knobs while also leveraging a variety

of benchmark circuit algorithms. Figure 5.1 gives a graphical representation of the design space and how other tool-flows fit into it.

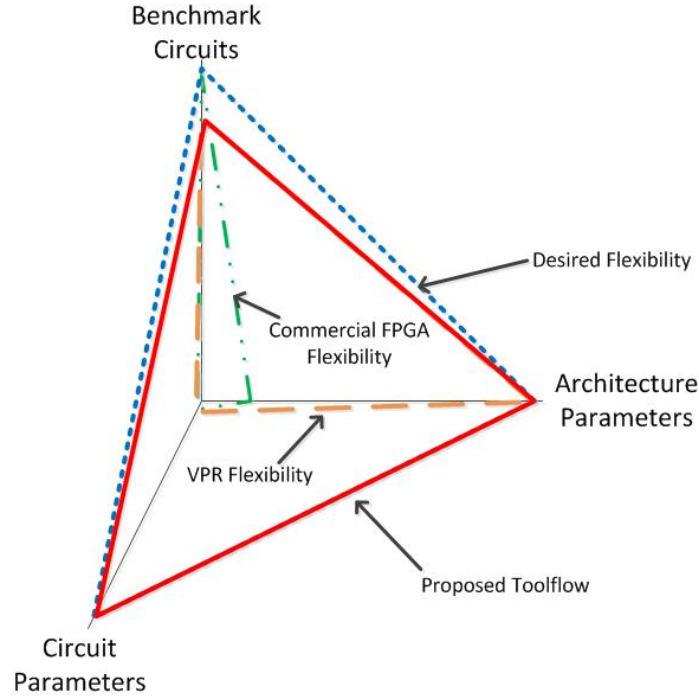


Figure 5.1: Graphical representation of FPGA hardware design space. Proposed tool-flow will address all three portions of the design space to move beyond other tools capabilities.

5.3 Research Question

What set of scripts is necessary for rapid configuration and creation of custom FPGA fabrics with different architectural and circuit parameters?

5.4 Approach

The proposed tool-flow will allow us to quickly generate full-FPGA schematics, configure them to perform any Verilog-based function, and prepare simulations to be run by the user to observe metrics of interest. This tool-flow will leverage the VTR tool to configure an FPGA with the architecture of our choosing, Cadence SKILL scripts to build schematics, and perl code to edit, generate, and control all of the information and tools used. Not only does this allow for rapid design space exploration, but it also allows for others who are not as fluent with FPGA design to try different topologies to see if a custom-FPGA would meet their specs. Figure 5.2 is a flow-chart describing the proposed toolflow at a high level.

There are two inputs for this tool-flow. The first is a description of both architectural and circuit-

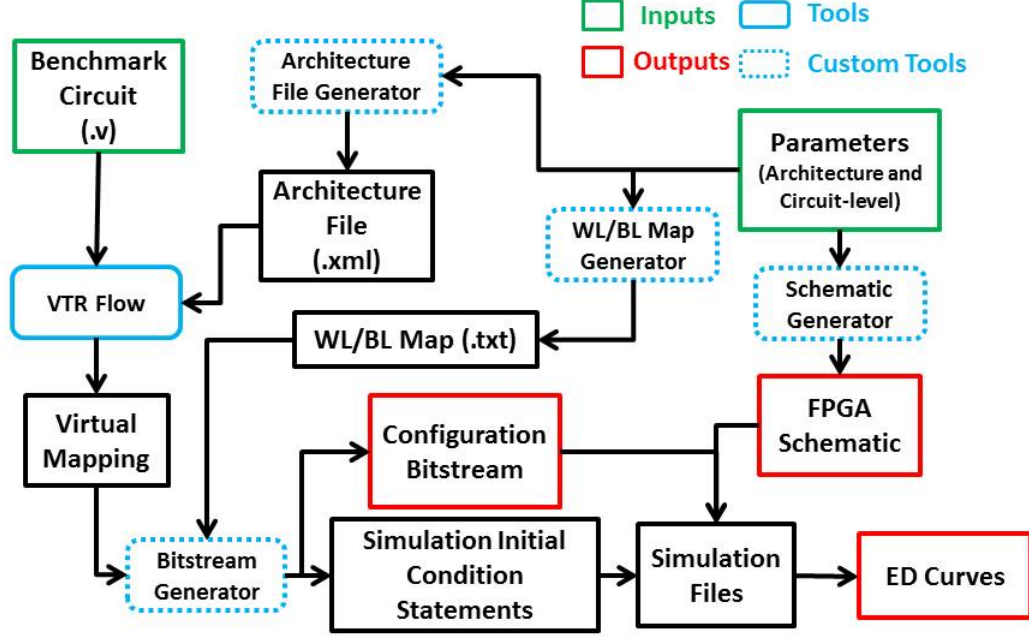


Figure 5.2: Graphical representation of FPGA hardware design space. Proposed tool-flow will address all three portions of the design space to move beyond other tools capabilities.

level parameters for the FPGA the researcher would like to build. Architectural parameters include things like channel width, LUT size, CLB clustering, and segment length, among others. Circuit-level parameters refer to routing switch topology, sizing, intra-CLB connectivity, etc. The second input to the toolflow is the benchmark circuit, or the functionality for the FPGA to perform. This tool-flow requires that the benchmark circuit be written in Verilog, as it is a requirement for the VTR tool. The Verilog-To-Routing (VTR) tool-flow allows users to see how a benchmark circuit would be implemented on a given architecture [?]. It has basic assumptions for circuit level parameters, which also allows the tool to make estimates of power, performance and area. For the purposes of this particular tool-flow, those estimates are not used, since we have the ability to generate our own schematics, which provide a more accurate representation of the specific structures. As inputs, the VTR flow takes a benchmark circuit (written in Verilog) and an architecture file, describing the overall architecture of the FPGA to be written to. The VTR outputs that are important for the

tool-flow, and the important information in them, are listed below.

- *Netlist file* - This file describes the connectivity of all of the blocks in the FPGA. The proposed tool-flow uses this file to determine intra-CLB connectivity.
- *BLIF file* - This file represents the logic in each LUT of the FPGA. This tool-flow takes the logic, which is represented in the Berkeley Logic Interchange Format (BLIF), and converts it into a binary representation, to be configured in the LUTs of the FPGA.
- *Placement File* - This file gives the relative X-Y locations of all of the logic, IP, and IO blocks mapped by the VTR tool. This tool-flow uses this script to keep track of all of the different configuration bits, as they are organized by the location of their FPGA-tile location.
- *Routing File* - This file describes which pins of different blocks are connected to each other through the global interconnect of the FPGA. The tool-flow uses this file to determine what values to set each configuration bit in the global interconnect.

We developed a set of custom scripts to parse the parameters file and generate important files. Each of these custom tools is described below.

- *Architecture File Generator* - This Perl takes as inputs the architectural (and some circuit-level) parameters and creates an architecture file (in .xml format) which is one of the inputs for the VTR tool-flow.
- *WL/BL Map Generator* - To be able to configure the FPGA properly, we need to understand the location in the fabric of each configuration bit throughout the FPGA fabric. This Perl script creates a map of where the bits will be located in the schematic/layout of the FPGA, given certain architectural and circuit-level parameters. It does this hierarchically, starting with small FPGA sub-circuits, and finally tiling them all together. This script also assumes an SRAM-like configuration style, placing each configuration bit within a large matrix of word lines (WLs) and bit lines (BLs).
- *Schematic Generator* - Using a similar methodology, this SKILL script creates schematics in the Cadence environment for FPGA sub-circuits, and then finally the full FPGA, from the architectural and circuit-level parameters provided. It is important for the locations of configuration bits in the schematic generator to match exactly with those from the WL/BL map generator

- *Bitstream Generator* - In order to truly test the functionality of the FPGA, a circuit has to be mapped to it, which we can then simulate or power up in hardware. To do this, we develop a set of Perl scripts that takes the virtual mapping given from the VTR tool-flow and determines which physical configuration bits need to be set to 1 and 0 in the FPGA fabric we have built by using the WL/BL map generated earlier. This script will then provide both a bitstream for configuring an FPGA through the scan-chain configuration scheme we implement on our FPGAs, and additionally provide a set of initial condition statements, which configure the FPGA for simulation. Thus, the tool-flow allows researchers to simulate the configuration of the FPGA or bypass it and go straight to simulating the functionality of the FPGA.

5.5 Results

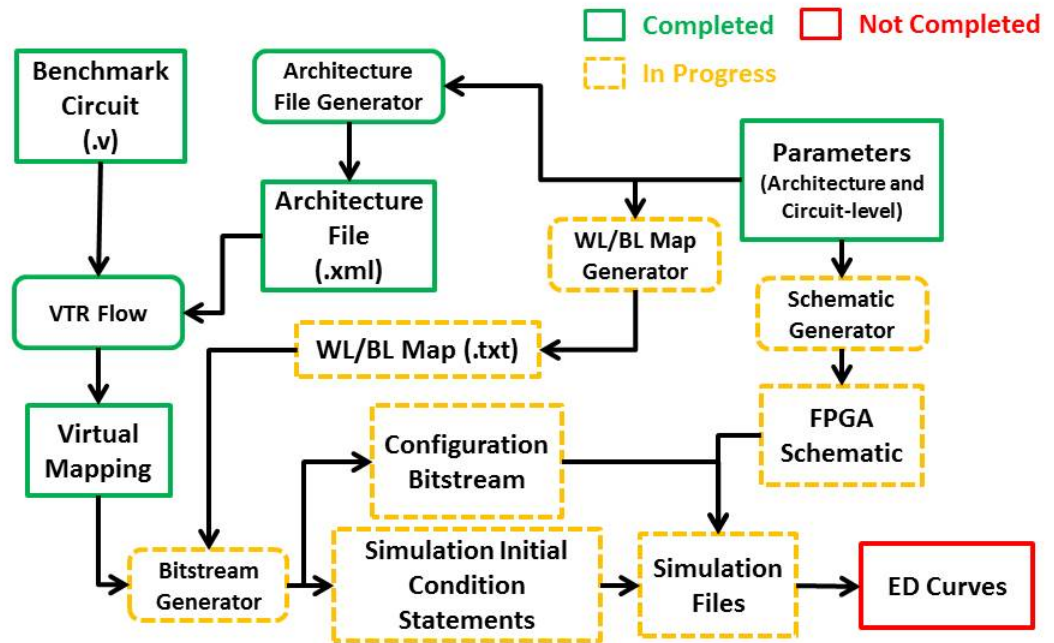


Figure 5.3: Progress on the tool-flow

Figure 5.3 shows a color-coded form of the tool-flow, illustrating progress on the individual portions of the toolflow. At this time, the only frozen portions (completely done with no need for

editing) are the VTR tool-flow and the architecture generation. This dissertation will not do any work on the WL/BL mapping script (that is another researchers work), but the schematic generation uses similar coding structures, in order for the configuration bit locations to match. At this point, the following sub-circuits can be generated and used for simulation purposes (based on the provided architectural and circuit-level parameters):

- Look-Up Tables
- Basic Logic Elements
- Connection boxes
- Switch points
- Switch box
- Intra-CLB interconnect multiplexers
- CLB tiles (which include 1 BLE and all of the necessary intra-CLB connectivity associated)
 - Mux based and Mini-FPGA style
- Full CLBs
 - Mux based and Mini-FPGA style
- FPGA Tile (CLB with connection boxes and one switch box)

The bitstream generator, though marked as work in progress, is essentially completed. The mechanisms for generating the different outputs (configuration bitstream in initial condition statements) are completed, but naming conventions for the different blocks have changed. The update for this script is simply making sure all of the bitcell descriptions and sub-circuit names are updated.

5.6 Anticipated Results

For this dissertation, the toolflow will be completed. To complete the schematic generation script, SKILL script generation circuits still need to be built for the following sub-circuits:

- IO-Block
- Full FPGA

All of the schematic generation properly generates the connectivity between the signals of the different sub-circuits in the FPGA. Unfortunately, the WL/BL allocation so far is not completely flexible. For small k and N , the WLs and BLs are allocated properly. As k and N increase, however, the code written starts to try and use WLs and BLs that are not allocated for the block. Once this issue is fixed, the schematics will be able to be simulated using both initial condition commands and the configuration bit stream. Because the SKILL code for the schematic generation is built hierarchically, it makes it easy to introduce new knobs if an FPGA designer wants to. For example, it will be very easy to extend the code to be flexible along bitcell topologies as well, because that only needs to be added to the lower-level blocks (like LUTs and switch points), and the changes will propagate to the top level of the FPGA. Finally, once all of the individual scripts are completed, it will be important to create a user-friendly interface which combines all of these smaller scripts together. The goal is to simply have two inputs, the parameter file and the Verilog circuit, and get ED curves at the output, showing how the generated FPGA performs the prescribed function across supply voltage. The dissertation will also provide more descriptive pseudo-code for the functionality of each of the smaller scripts of the toolflow.

5.7 Contributions

The contributions for this section are as follows:

- Development of toolflow to generate FPGA fabrics for rapid design space exploration of circuit-level and architectural parameters
- Easy-to-use tool-flow for researchers who want to build an FPGA to observe how the FPGA circuitry will operate pre-fabrication
- Generation of configuration bitstreams and initial condition statements for FPGA simulations from VTR outputs
- Circuit-level and architectural parameter co-optimization for ULP FPGA hardware design

6 Embedded FPGAs in SoCs

6.1 Motivation

A popular solution for ULP wireless sensing is developing ASIC-based SoCs with a low-power controller. An example system is introduced in [1]. This SoC is designed to run off of harvested energy, and doesn't even need a battery. While the extremely low power consumption is state-of-the-art, there are still potential issues with the design. First, the node is prohibitively inflexible for widespread adoption. Right now, this node can only be used for ECG sensing applications. A node like this would be a viable solution for all IoT applications, but the cost of re-designing the SoC to tailor its design to each application (100s potentially) is extreme, both in time and monetarily. Additionally, making any updates or changes to the algorithms implemented in any of these SoCs also requires a respin of the chip. Aside from flexibility, testing ASIC SoCs can be difficult and costly. Introducing test structures into the different ASIC blocks of the SoC can hinder block performance, and it's difficult to attain adequate observability and testability in large, complicated SoCs. Including small FPGA fabrics, designed for ULP functionality, could potentially revolutionize ULP sensing. Functions that are common to most sensing applications (filtering, arithmetic operations, etc) should remain ASIC-implemented, as ASICs are the most power-efficient implementation of algorithms. Algorithms that are prone to change, or those that are specific to different applications, should be implemented in the low-power reconfigurable fabric in the SoC. That way, a single SoC could be retargeted for a host of different applications, and the only necessary change would be a re-configuration of the FPGA-portion of the SoC to the specific application. Including reconfigurable fabric on SoCs provides additional benefits other than increasing flexibility. The FPGA fabric can be configured as test structures for both itself and the entire SoC, allowing built-in self-test (BIST) for the SoC with greatly reduced overhead. This enables the removal of test structures from the ASIC blocks, enhancing their efficiency.

6.2 Prior Art

[4] discusses the benefits of embedding FPGA fabrics into SoCs. This paper gives a methodology for using the FPGA fabric to test itself, and also use it as a means of testing other cores in the SoC. Leveraging the techniques described in this paper would allow a ULP sensor to be tested with as little cost and overhead as possible. Commercial FPGA companies have seen the potential benefits of integrating FPGA fabrics into SoCs, and have begun developing FPGA-based SoCs themselves.

Xilinx Zynq 7000 All Programmable SoC leverages their state-of-the-art FPGA technology (either Artix-7 or Kintex-7) in combination with an ARM Dual Core processor and a variety of ASIC blocks (DSP slices, floating point processor extension, caches, etc.) [15]. Altera has its own set of FPGA-based SoCs, the newest of which is the Arria 10. This device uses the same ARM processor, but boasts additional security measures, and leverages Altera FPGA technology [16]. These devices, however, are targeted for high-power, high-performance applications, where GHz speeds are necessary. As a result, the power consumption of these devices is on the order of Ws. ULP applications need the power consumption to be in the μW range. Because commercial FPGA companies have illustrated that SoCs with FPGA-fabrics are viable solutions, re-targeting the designs for low-power seems viable.

6.3 Research Question

Can embedding FPGA fabric in ULP SoCs improve flexibility while keeping the power consumption low enough to maintain ULP functionality, allowing ULP operation across a larger class of applications?

6.4 Approach

The first step in this research thrust will be to determine what algorithms and tasks in an ULP SoC to be implemented in FPGA fabrics. To do this, we will survey the different ASIC blocks of the SoC described in [1]. From first glance, it seems as though the main algorithms that seem specific to that target application are AFIB/R-R detection and possible envelope detection, although envelope detection could be a common algorithm among multiple sensing applications. It will also be important to attain a better understanding of the things in that SoC implementation that are prone to change, and implement those algorithms on low-power FPGA fabrics as well. Next, it will be important to optimize circuit elements and architectures for inclusion of FPGA fabric in ULP SoCs. Circuit elements will be optimized in chapter 2. Architectures are optimized in chapter 3. It is probable that the same circuit optimizations for ULP FPGAs will be employed in the embedded FPGA fabric. Because the range of algorithms for the SoC is limited compared to the ones explored in chapters 2-4, it is possible for the optimal architecture for the reconfigurable portion of the SoC to be different. Thus, it will be important to optimize the design within a more targeted design space. The toolflow discussed in chapter 4 will be leaned on heavily for this design space exploration. Once the reconfigurable block for the SoC is designed, we will configure and simulate

the reconfigurable logic and compare the results to the ASIC implementations. Its probable that the FPGA implementation of some of these algorithms will consume more power than the ASIC implementation, but that doesnt make it useless. If the FPGA implementation, along with the rest of the SoC, falls within the stringent power budgets of low-power applications, then the FPGA fabric allows the SoC, which before was only functional for a single application, to be target to a host of new applications, orienting ULP SoC development into a direction that is viable for large-scale ULP applications like the IoT.

6.5 Evaluation Metrics

We will determine whether the inclusion of ULP reconfigurable fabric is viable for ULP sensing SoCs through evaluation of the following metrics:

- **FPGA Size** - Devices for ULP applications have strict size restrictions, for comfortability and aesthetics in implementation. While FPGA fabrics can implement any function, given enough logic area, there is a practical limit to the number of FPGA logic resources that can be available for an FPGA block on an SoC targeted for ULP applications. It will be important to make sure that the algorithms partitioned for the FPGA fabric can be implemented using relatively few logic blocks.
- **Power Consumption** - It will be important to characterize the power consumption of the reconfigurable logic block as it performs the different functions partitioned for it.
- **Testability** - In order to further incentivize the use of FPGA fabrics in ULP SoCs, showing how the FPGA fabric can be used to test an ULP SoC in a specific test case will be a proof of concept, further showing the flexibility of the FPGA fabric. The testability of the FPGA fabric, however, is directly proportional to the size of the FPGA fabric, so this metric and the FPGA size metric oppose each other, and must be co-optimized
- **Energy Consumption** - While it is important for ULP SoCs to have low power consumption, so as to be able to use smaller and smaller power sources, it is also important for the circuit to have low energy consumption, so as to extend the lifetime of the power source. Therefore, it will be important to make sure that the energy consumption of the reconfigurable logic is still low enough for inclusion in the ULP SoC.

6.6 Anticipated Results

Using the circuit optimizations from chapter 2, and retargeting the circuit-architecture co-optimization using the tool-flow created in chapter 4 for a smaller set of potential, application-specific algorithms, an FPGA fabric will be designed for inclusion in an ULP SoC. Hopefully, the fabric will have low enough area, power consumption, and energy consumption to be included in ULP SoCs, thus enabling the device to be both tested more efficiently and have more wide-spread efficacy. If this is true, then this new SoC can be retargeted for a large variety of sensing applications. It is possible, however, for any of these metrics to be too high, as FPGA fabrics have yet to have such low power consumptions. This study is more to observe and understand the feasibility of creating ULP SoCs with reconfigurable portions.

6.7 Contributions

The expected contributions for this section are as follows:

- Body Sensor Node algorithm implementations in ULP FPGA fabric
- Comparison between ASIC and FPGA implementations of BSN algorithms
- Exploration into the feasibility of FPGA implementations for ULP wireless sensing
- Creation and implementation of test structures for BSN algorithms in FPGA fabric

7 Research Tasks and Schedule

Subject	#	Task Description	Status	Related Publications
Circuit-Level Optimization	1	Initial Routing Switch Exploration	Completed	[OAA1]
	2	Initial Sense Amp Exploration	Completed	
	3	CLB Simulations	February '15	[OAA2]
	4	5T Bitcell Testing	March '15	
	5	Revisited Routing Switch Simulations	April '15	
	5	Configuration Bit Simulations	May '15	
	6	Small FPGA/Test Structure Tapeout	Summer '15	
	7	Chip Testing	December '15	[OAA3] [OAA4]
Architectural Optimization	1	Initial Architecture Exploration	Completed	
	2	Characterization of FPGA sub-circuits for VPR Power Optimizations	Aug '15	[OAA5]
	3	FPGA Architecture Design Space Exploration (through VPR)	Sept '15	
	4	FPGA Architecture Comparison through Simulation	Oct '15	[OAA6]
Custom Toolflow	1	Finish Schematic Generation Script	Feb '15	[OAA7]
	2	Fully update bitstream/IC generator	Feb '15	
	3	Overall toolflow wrapper	March '15	
	4	Proof of concept simulations	April '15	
Embedded FPGA Fabric	1	Determine algorithms for embedded FPGA fabric	Dec '15	
	2	Comparison of FPGA implementations to ASIC implementations	Jan '16	[OAA8]
	3	Feasibility analysis of FPGA Implementations of BSN algorithms	Feb '16	[OAA9]

8 Publications

8.1 Completed

- [OAA1] – Oluseyi A. Ayorinde and Benton H. Calhoun. 2013. Circuit optimizations to minimize energy in the global interconnect of a low-power-FPGA (Poster). In Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays (FPGA '13). ACM, New York, NY, USA, 277-277.

8.2 Planned

- [OAA2] - Using FPGA-style Local Interconnect in CLBs for Low-Power FPGAs
- [OAA3] - Exploring routing switch topologies for ULP FPGA interconnects
- [OAA4] - Configuration Bits for ULP FPGAs
- [OAA5] - Re-evaluating architectures for Sub-mW FPGAs
- [OAA6] - A new architecture for Sub-mW FPGAs (could be combined w/ [OAA5])
- [OAA7] - RGC: A toolflow for generating custom FPGA schematics and configurations
- [OAA8] - Comparison of IoT algorithms on Sub-mW FPGAs vs. ASICs
- [OAA9] - Feasibility Analysis of Embedded FPGAs for Sub-mW SoCs

Bibliography

- [1] Fan Zhang; Yanqing Zhang; Silver, J.; Shakhsheer, Y.; Nagaraju, M.; Klinefelter, A.; Pandey, J.; Boley, J.; Carlson, E.; Shrivastava, A.; Otis, B.; Calhoun, B., "A batteryless 19W MICS/ISM-band energy harvesting body area sensor node SoC," Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International , vol., no., pp.298,300, 19-23 Feb. 2012
- [2] E. Ahmed and J. Rose, *The effect of LUT and cluster size on deepsubmicron FPGA performance and density*. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Vol. 12, No. 3, pp 288–298, March, 1994.
- [3] Fei Li, Deming Chen, Lei He, and Jason Cong. 2003. Architecture evaluation for power-efficient FPGAs. In Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays (FPGA '03). ACM, New York, NY, USA, 175-184.
- [4] Abramovici, M.; Stroud, C.; Emmert, M., "Using embedded FPGAs for SoC yield improvement," Design Automation Conference, 2002. Proceedings. 39th , vol., no., pp.713,724, 2002
- [5] Jamieson, P.; Luk, W.; Wilton, S.J.E.; Constantinides, G.A., "An energy and power consumption analysis of FPGA routing architectures," Field-Programmable Technology, 2009. FPT 2009. International Conference on , vol., no., pp.324,327, 9-11 Dec. 2009
- [6] Ryan, J.F.; Calhoun, B.H., "A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS," *Custom Integrated Circuits Conference (CICC)*, 2010 IEEE , vol., no., pp.1,4, 19-22 Sept. 2010
- [7] Grossmann, P.J.; Leeser, M.E.; Onabajo, M., "Minimum Energy Analysis and Experimental Verification of a Latch-Based Subthreshold FPGA," *Circuits and Systems II: Express Briefs, IEEE Transactions on* , vol.59, no.12, pp.942,946, Dec. 2012
- [8] Tuan, T.; Rahman, A.; Das, S.; Trimberger, S.; Sean Kao, "A 90-nm Low-Power FPGA for Battery-Powered Applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.26, no.2, pp.296,300, Feb. 2007
- [9] Anderson, J.H.; Najm, F.N., "Low-Power Programmable FPGA Routing Circuitry," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.17, no.8, pp.1048,1060, Aug. 2009

- [10] Guy Lemieux and David Lewis. 2001. Using sparse crossbars within LUT. In Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays (FPGA '01), Martine Schlag and Russell Tessier (Eds.). ACM, New York, NY, USA, 59-68.
- [11] Microsemi Corporation, "IGLOO nano FPGA Fabric (User's Guide)," Version 1.4, March 2008 [Revised October 2012].
- [12] Lattice Semiconductor, iCE40 Ultra Family Data Sheet, DS1048 Version 1.5 datasheet, Oct. 2014.
- [13] Texas Instruments, MSP430F21x1 Mixed Signal Microcontroller, SLAS439F datasheet, Sept. 2004 [Revised Aug. 2011].
- [14] Xilinx Corporation, "Applications," <http://www.xilinx.com/applications.html>
- [15] Xilinx Corporation, "Zynq-7000 All Programmable SoC Overview," DS190 (v1.7) datasheet, Oct. 2014.
- [16] Altera Corporation, "Arria 10 Device Datasheet," datasheet, Jan. 2015.
- [17] SourceTech411, "Top FPGA Companies for 2013" <http://sourcetech411.com/2013/04/top-fpga-companies-for-2013/>
- [18] Jason Luu, Jeffrey Goeders, Michael Wainberg, Andrew Somerville, Thien Yu, Konstantin Nasartschuk, Miad Nasr, Sen Wang, Tim Liu, Nooruddin Ahmed, Kenneth B. Kent, Jason Anderson, Jonathan Rose, and Vaughn Betz. 2014. "VTR 7.0: Next Generation Architecture and CAD System for FPGAs." *ACM Trans. Reconfigurable Technol. Syst.* 7, 2, Article 6 (July 2014), 30 pages.